INTEGRATION, the VLSI journal 44 (2011) 305–315

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decoder

Yang Sun *, Joseph R. Cavallaro

Department of Electrical and Computer Engineering, Rice University, 6100 Main Street, Houston, TX 77005, USA

ARTICLE INFO

Available online 17 July 2010

Keywords:
QPP interleaver
Quadratic permutation polynomial
Turbo decoder
MAP decoder
VLSI
ASIC
3GPP LTE

ABSTRACT

We present an efficient VLSI architecture for 3GPP LTE/LTE-Advance Turbo decoder by utilizing the algebraic-geometric properties of the quadratic permutation polynomial (QPP) interleaver. The high-throughput 3GPP LTE/LTE-Advance Turbo codes require a highly-parallel decoder architecture. Turbo interleaver is known to be the main obstacle to the decoder parallelism due to the collisions it introduces in accesses to memory. The QPP interleaver solves the memory contention issues when several MAP decoders are used in parallel to improve Turbo decoding throughput. In this paper, we propose a low-complexity QPP interleaving address generator and a multi-bank memory architecture to enable parallel Turbo decoding. Design trade-offs in terms of area and throughput efficiency are explored to find the optimal architecture. The proposed parallel Turbo decoder has been synthesized, placed and routed in a 65-nm CMOS technology with a core area of 8.3 mm^2 and a maximum clock frequency of 400 MHz. This parallel decoder, comprising 64 MAP decoder cores, can achieve a maximum decoding throughput of 1.28 Gbps at 6 iterations

1. Introduction

3GPP Long Term Evolution (LTE) [1], which is a set of enhancements to the 3G Universal Mobile Telecommunications System (UMTS) [2], has received tremendous attention recently and is considered to be a very promising 4G wireless technology. For example, Verizon Wireless has decided to deploy LTE in their next generation 4G evolution. One of the main advantages of 3GPP LTE is high throughput. For example, it provides a peak data rate of 326.4 Mbps for a 4×4 antenna system, and 172.8 Mbps for a 2×2 antenna system for every 20 MHz of spectrum. Furthermore, LTE-Advance [3], the further evolution of LTE, promises to provide up to 1 Gbps peak data rate.

The channel coding scheme for LTE is Turbo coding [4]. The Turbo decoder is typically one of the major blocks in a LTE wireless receiver. Turbo decoders suffer from high decoding latency due to the iterative decoding process, the forward–backward recursion in the maximum *a posteriori* (MAP) decoding algorithm and the interleaving/de-interleaving between iterations [5–7]. Generally, the task of an interleaver is to permute the soft values generated by the MAP decoder and write them into random or pseudo-random positions.

A high throughput Turbo decoder can be realized by parallelizing several MAP decoders, where each MAP decoder operates on a segment of the received codeword [8]. Due to the randomness of the Turbo interleaver, two or more MAP decoders may access the same memory at the same clock cycle which will lead to a memory collision. As a result, the decoder has to be stalled which consequently delays the decoding process. The Interleaver structures in the current 3G standards, such as CDMA/W-CDMA/UMTS, do not have a parallel structure. Although the memory stalls caused by the interleaver can be partially reduced by using write buffers [9], the memory stalls will become more and more frequently as the parallelism degree increases. To solve this problem, the high data rate 3GPP LTE standard has adopted a contention-free, parallel interleaver which is called quadratic permutation polynomial (QPP) Turbo interleaver [4]. From an algebraic-geometric perspective, the QPP interleaver allows analytical designs and simplifies hardware implementation of a parallel Turbo decoder [10]. Based on the permutation polynomials over integer rings, every factor of the interleaver length can be a parallelism degree for the decoder [10] which is contention-free.

In the literature, many decoder architectures have been extensively investigated for 3G or 3G-like Turbo codes [11–18]. Recently, several high speed Turbo decoders have been developed for 3GPP LTE standard [19–22]. As a 4G candidate system, the 3GPP LTE-Advance system is pushing for 1 Gbps data rate. Thus, it is very important and challenging to design a Turbo decoder to support such a high data rate. In this paper, we propose an efficient hardware architecture for 3GPP LTE/LTE-Advance Turbo decoder. A low-complexity circuit is designed to generate the QPP interleaving addresses on the fly. By utilizing the QPP contention-free property, memory systems are partitioned into multiple banks to allow concurrent accesses by multiple MAP decoders. More than 1 Gbps data rate is feasible with the proposed decoding scheme.

* Corresponding author.
E-mail addresses: ysun@rice.edu (Y. Sun), cavallar@rice.edu (J.R. Cavallaro).

0167-9260/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2010.07.001

The rest of this paper is organized as follows. Section 2 reviews the fundamentals of Turbo codes. Section 3 describes the basic structure of the QPP interleaver and its several algebraic properties. Then we propose an online address generator for the QPP interleaver. In Section 4, two types of low-latency MAP decoder architectures are introduced and compared. By employing multiple MAP decoder cores and multiple QPP interleavers, we present a parallel Turbo decoder architecture in Section 5. Then the VLSI implementation results are summarized and compared with existing Turbo decoders.

2. Fundamentals of turbo codes

In order to explain the proposed parallel Turbo decoder architecture, the fundamentals of Turbo codes are briefly described in this section.

2.1. Turbo encoder structure

As shown in Fig. 1, the Turbo encoding scheme in the LTE standard is a parallel concatenated convolutional code with two 8-state constituent encoders and one quadratic permutation polynomial (QPP) interleaver [4]. The function of the QPP interleaver is to take a block of N-bit data and produce a permutation of the input data block. From the coding theory perspective, the performance of a Turbo code depends critically on the interleaver structure [23]. The basic LTE Turbo coding rate is 1/3. It encodes an N-bit information data block into a codeword with $3N+12$ data bits, where 12 tail bits are used for trellis termination. The initial value of the shift registers of the 8-state constituent encoders shall be all zeros when starting to encode the input information bits. LTE has defined 188 different block sizes, $40 \le N \le 6144$.

2.2. Turbo decoder structure

The basic structure of a Turbo decoder is functionally illustrated in Fig. 2. A Turbo decoder consists of two maximum *a posteriori* (MAP) decoders [24,25] separated by an interleaver that permutes the input sequence. The decoding is an iterative process in which the so-called extrinsic information is exchanged between MAP decoders. Each Turbo iteration is divided into two half iterations. During the first half iteration, MAP decoder 1 is enabled. It receives the soft channel information (soft value L_s for the systematic bit and soft value $L_{p\ 1}$ for the parity bit) and the *a priori* information L_a^1 from the other constituent MAP decoder through deinterleaving (π^{-1}) to generate the extrinsic information L_e^1 at its output. Likewise, during the second half iteration, MAP decoder 2 is enabled, and it receives the soft channel information (soft value L_s for a permuted version of the systematic bit and soft value $L_{p\ 2}$ for the parity bit) and the *a priori* information L_a^2 from

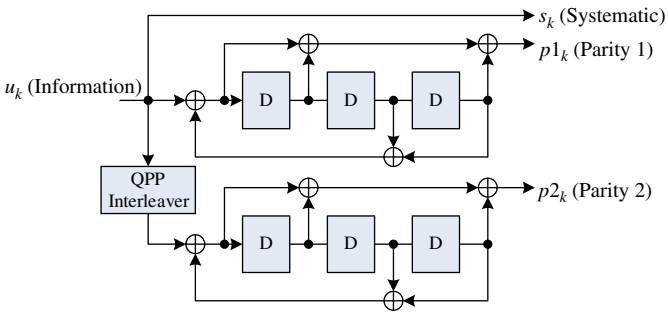

Fig. 1. Structure of rate 1/3 Turbo encoder in LTE.

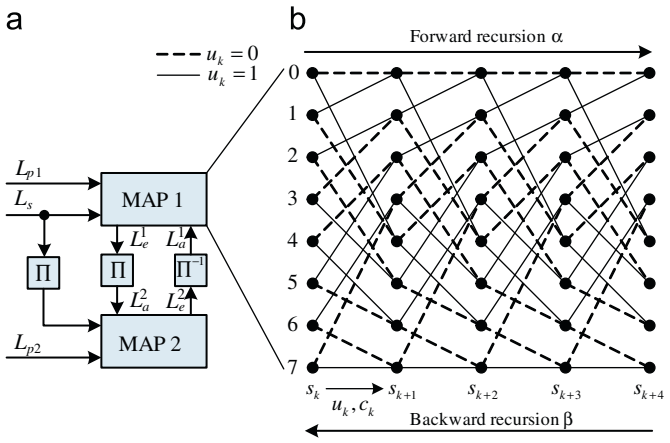

Fig. 2. Basic structure of an iterative Turbo decoder. (a) Iterative decoding based on MAP decoders. (b) Forward/backward recursions on the trellis diagram.

MAP decoder 1 through interleaving (π) to generate the extrinsic information L_e^2 at its output. This iterative process repeats until the decoding has converged or the maximum number of iterations has been reached.

The MAP algorithm at each constituent MAP decoder computes the log-likelihood ratios (LLRs) of the *a posteriori* probabilities (APPs) for information bit u_k as follows: [7,26]

$$\mathrm{LLR}(\hat{u}_k) = \max_{\mathbf{u}:u_k=1}^{*}\{\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1},s_k) + \beta_k(s_k))\} - \max_{\mathbf{u}:u_k=0}^{*}\{\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1},s_k) + \beta_k(s_k))\}, \quad (1)$$

where α_k and β_k denote the forward and backward state metrics, and are recursively computed as follows:

$$\alpha_k(s_k) = \max_{s_{k-1}}^{*}\{\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1},s_k)\}, \quad (2)$$

$$\beta_k(s_k) = \max_{s_{k+1}}^{*}\{\beta_{k+1}(s_{k+1}) + \gamma_k(s_k,s_{k+1})\}. \quad (3)$$

The γ_k term above is the branch transition probability that depends on the trellis diagram, and is usually referred to as the branch metric. The max star operator employed in the above descriptions is defined as follows: [26]

$$\max^{*}(a,b) = \log(e^a + e^b) = \max(a,b) + \log(1 + e^{-|a-b|}). \quad (4)$$

3. QPP interleaver

Interleaving/deinterleaving of extrinsic information is a key issue that needs to be addressed to enable parallel decoding because memory access contention may occur when MAP decoders fetch/write extrinsic information from/to memory. The QPP interleaver defined in the new 3GPP LTE standard differs from previous 3G interleavers in that it is based on algebraic constructions via permutation polynomials over integer rings. It is known that permutation polynomials generate contention-free interleavers [27,10], i.e. every factor of the interleaver length becomes a possible parallelism degree.

3.1. Algebraic description of QPP interleaver

The QPP interleaver can be expressed via a simple mathematical formula. Given an information block length N, the x-th interleaving output position is specified by the quadratic

expression: [4]

$$f(x) = (f_2x^2 + f_1x) \bmod N, \tag{5}$$

where parameters f_1 and f_2 are integers and depend on the block size N ($0 \le x, f_1, f_2 < N$). For each block size, a different set of parameters f_1 and f_2 are defined. In LTE, all the block sizes are even numbers and are divisible by 4 and 8. Moreover, the block size N is always divisible by 16, 32, and 64 when $N \ge 512$, 1024, and 2048, respectively. By definition, parameter f_1 is always an odd number whereas f_2 is always an even number. Through further inspection, we can list the following algebraic properties for the QPP interleaver.

QPP interleaver algebraic property 1: $f(x)$ has the same even/odd parity as x:

$$f(2k) \bmod 2 = 0$$

$$f(2k+1) \bmod 2 = 1.$$

QPP interleaver algebraic property 2: The remainders of $f(x)/4$, $f(x+1)/4$, $f(x+2)/4$, and $f(x+3)/4$ are unique:

$$f(4k) \bmod 4 = 0$$

$$f(4k+1) \bmod 4 = \begin{cases} 1 & \text{when } (f_1+f_2) \bmod 4 = 1 \\ 3 & \text{when } (f_1+f_2) \bmod = 3 \end{cases}$$

$$f(4k+2) \bmod 4 = 2$$

$$f(4k+3) \bmod 4 = \begin{cases} 3 & \text{when } (f_1+f_2) \bmod 4 = 1 \\ 1 & \text{when } (f_1+f_2) \bmod 4 = 3. \end{cases}$$

QPP interleaver algebraic property 3:

$$f(x) \bmod n = f(x+m) \bmod n, \ \forall m : m \bmod n = 0.$$

Property 1 can be easily verified since parameter f_2 is always even and parameter f_1 is always odd by definition. Property 2 can be shown through the following equations:

$$f(4k) = 4(4f_2k^2 + f_1k)$$

$$f(4k+1) = 4(4f_2k^2 + 2f_2k + f_1k) + f_2 + f_1$$

$$f(4k+2) = 4(4f_2k^2 + 4f_2k + f_1k + f_2) + 2f_1$$

$$f(4k+3) = 4(4f_2k^2 + 6f_2k + f_1k + 2f_2) + f_2 + 3f_1.$$

Property 3 can be verified by

$$f(x+m) = f(x) + m(2f_2x + f_2m + f_1).$$

We will explain later that these algebraic properties are very useful in designing memory systems for parallel Turbo decoder.

3.2. QPP contention-free property

In general, a Turbo interleaver/de-interleaver $f(x)$, is said to be contention-free for a window size of L if and only if it satisfies the following constraint [10,28,29]

$$\left\lfloor \frac{f(x+iL)}{L} \right\rfloor \neq \left\lfloor \frac{f(x+jL)}{L} \right\rfloor, \tag{6}$$

where $0 \le x < L$, $0 \le i,j < P$ ($=N/L$), and $i \ne j$. The terms in (6) are essentially the memory indices that are concurrently accessed by the P MAP decoder cores. If these memory indices are unique during each read and write operation, then there are no contentions in memory accesses. Fig. 3 shows an example of the contention-free memory access scheme.

It has been shown in [27,10] that every factor of the interleaver length N becomes a possible interleaver parallelism that satisfies the contention-free requirement in (6). Table 1 summaries the parallelism degrees (up to 64) for some of the LTE QPP interleavers.

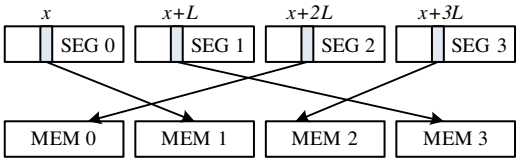

Fig. 3. An example of the contention-free interleaving, where a data block is divided into P=4 segments (SEG 0–SEG 3) with equal length of $L=N/P$. The contention-free property requires that for a fixed offset x at each segment, the segment indices for the interleaving addresses $\lfloor f(x+iL)/L \rfloor$ ($0 \le i \le P-1$) are unique so that they can be physically mapped to different memory modules.

Table 1
QPP interleaver parallelism.

N	$f(x)$	Parallelism (factors of N)
40	$10x^2+3x$	1,2,4,5,8,10,20
48	$12x^2+7x$	1,2,3,4,6,8,12,16,24
64	$42x^2+19x$	1,2,4,8,16,32
…	…	…
6016	$94x^2+23x$	1,2,4,8,16,32,47,64
6080	$190x^2+47x$	1,2,4,5,8,10,16,19,20,32,38,40,64
6144	$480x^2+263x$	1,2,3,4,6,8,12,16,24,32,48,64

3.3. Hardware implementation of QPP interleaver

Based on the algebra analysis in [27], the QPP interleaver is guaranteed to always generate a unique address which greatly simplifies the hardware implementation. In MAP trellis decoding, the QPP interleaving addresses are usually generated in a consecutive order (with step size of d). By taking advantage of this fact, the QPP interleaving address can be computed in a recursive manner. Suppose the interleaver starts at x_0, we first pre-compute $f(x_0)$ as

$$f(x_0) = (f_2x_0^2 + f_1x_0) \bmod N. \tag{7}$$

In the following cycles, as x is incremented by d, $f(x+d)$ is computed recursively as follows:

$$f(x+d) = (f_2(x+d)^2 + f_1(x+d)) \bmod N \tag{8}$$

$$= (f(x) + g(x)) \bmod N, \tag{9}$$

where $g(x)$ is defined as

$$g(x) = (2df_2x + d^2f_2 + df_1) \bmod N. \tag{10}$$

Note that $g(x)$ can also be computed in a recursive manner:

$$g(x+d) = (g(x) + 2d^2f_2) \bmod N \tag{11}$$

$$= (g(x) + (2d^2f_2 \bmod N)) \bmod N. \tag{12}$$

The initial value $g(x_0)$ needs to be pre-computed as

$$g(x_0) = (2df_2x_0 + d^2f_2 + df_1) \bmod N. \tag{13}$$

The modulo operation in (9) and (12) can be difficult to implement in hardware if the operands are not known in advance. However, by definition we know that both $f(x)$ and $g(x)$ are less than N so calculating (9) and (12) can be realized by additions. In the proposed method, three numbers need to be pre-computed: $(2d^2f_2) \bmod N$, $f(x_0)$, and $g(x_0)$. Fig. 4 shows a hardware architecture to compute the interleaving address $f(x)$, where x starts from x_0 and is incremented by d on every clock cycle. For example, by setting d to 1, this circuit can generate interleaving addresses at each step of 1. If n consecutive interleaving addresses

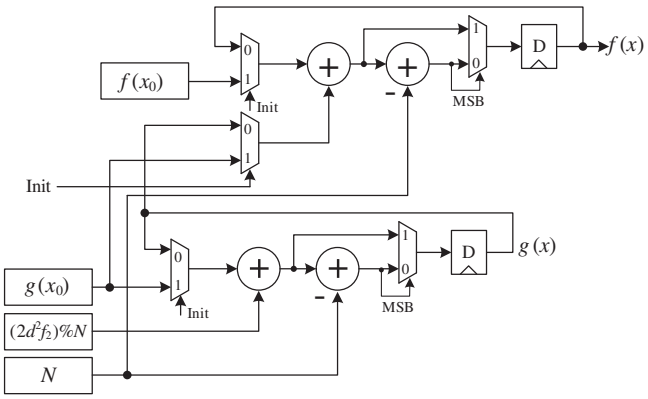

Fig. 4. Forward QPP address generator circuit diagram, step size $=d$.

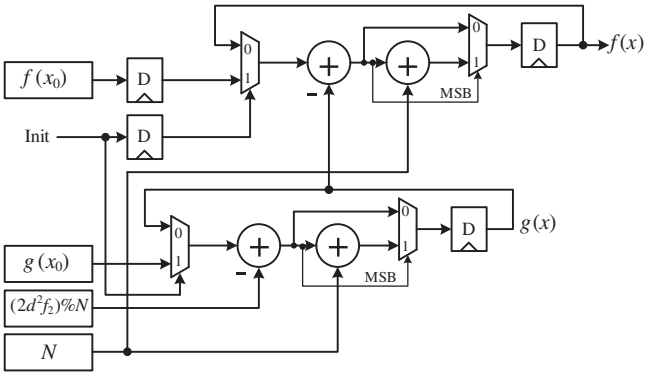

Fig. 5. Backward QPP address generator circuit diagram, step size $=d$.

are required at each clock cycle, this circuit can be replicated n times with n different initial values: x_0, $x_0+1,\ldots,$ and x_0+n-1.

The circuit in Fig. 4 can generate interleaving address in a descending order as well by setting d to be a negative number, eg. $d=-1$. But $g(x_0)$ needs to be recomputed for negative d. To be able to generate both forward and backward addresses using the same $f(x)$ and $g(x)$ functions, we now describe a method to generate the QPP interleaving address in the descending order. By substituting x with $x-d$ in (9) and reorganize (9), we can get

$$f(x-d) = (f(x)-g(x-d)) \bmod N. \tag{14}$$

Similarly, substitute x with $x-d$ in (12) and reorganize (12), we can get

$$g(x-d) = (g(x)-(2d^2 f_2 \bmod N)) \bmod N. \tag{15}$$

Based on (14) and (15), Fig. 5 shows a hardware architecture to compute the QPP address $f(x)$ in the descending order (backward generating), where x starts from x_0 and is decremented by d on every clock cycle. The three pre-computed values are the same as those in the forward QPP address generator (cf. Fig. 4).

As can be seen from Figs. 4 and 5, the proposed QPP interleaver pattern generator consumes very few resources. The complexity of this circuit is an order of magnitude smaller than the previous 3G interleavers. For example, a circuit with about 30 K gate count is reported in [30] to generate the interleaving addresses for Turbo codes in the previous 3G standard (3GPP Release-4), and a UMTS hardware interleaver with 10.5 K gate count is presented in [31].

4. MAP decoder architecture for LTE turbo codes

MAP decoder architectures have been studied by many researchers [24,25,32–35]. Several factors, such as interleaver structure and sliding window scheme, must be considered when choosing an appropriate MAP decoder for LTE Turbo decoding. In this section we modify two low-latency MAP decoder architectures and propose a low-complexity QPP interleaving address generator to operate full-speed with the MAP decoder.

Due to the double recursion in the MAP decoding algorithm [7], the MAP decoder suffers from high decoding latency. To reduce the decoding latency, the sliding window algorithm is often used [36]. However, the problem of the sliding window approach is the unknown backward (or forward) state metrics which are required in the beginning of the backward (or forward) recursion. We refer to the state metrics at sliding window length distance as *stakes*. These stakes can be estimated by using a training calculation [36], which will result in an additional decoding delay depending on the training length. For LTE Turbo codes, we do not recommend this traditional sliding window method when the Turbo coding rate is high. Because many parity bits will be removed after the base Turbo code is punctured to a higher code rate, the training length has to be increased to accurately estimate the state metrics at those stakes which consequently delays the decoding process.

For LTE Turbo decoding, we suggest to use a low-latency decoding method, referred to as state metric propagation (SMP) method, where the state metrics at stakes are initialized with stakes from the previous iteration [37]. In the very first iteration, uniform state metrics can be used for initialization. This method avoids the training calculation by propagating the state metrics to the next iteration. This method is especially useful when the Turbo coding rate is high. Based on our simulation results, the performance degradation caused by the window truncation in the SMP method is smaller than that in the traditional training based sliding window method in the case of high Turbo code rate. To compare the decoding performance using these two sliding window algorithms for high rate LTE Turbo codes, we perform floating point simulations using BPSK modulation over AWGN channel. The LTE rate matching algorithm [4] is used for code puncturing. Fig. 6 shows the floating-point simulation result for a rate of 0.95 Turbo code. Because of the high code rate, the maximum number of iterations is set to 10. In the figure, we show the block error rate (BLER) curves for the SMP based sliding window algorithm and the traditional training based sliding window algorithm. In the traditional training algorithm, we assume the training length is equal to the window length. As can be seen, the BLER performance of the SMP algorithm with window length $W=64$ is better than that of the training algorithm with window length $W=64$, and is close to that of the training algorithm with $W=96$. The SMP algorithm with $W=96$ and the training algorithm with $W=128$ perform close to the optimal case when there is no window effect. Because of the good decoding performance and low decoding delay, we adopted the SMP algorithm in our Turbo decoder design.

The SMP based sliding window (SW) MAP algorithm (SW-MAP) has a window overhead of W (c.f. Fig. 7(a)), which will lead to additional decoding delays. To eliminate this window overhead, we also consider a non-sliding window (NSW) based MAP algorithm (NSW-MAP) which is shown in Fig. 7(b). To be more general, we consider the case of decoding a segment of the code block where the segment length is $L=N/P$. In the SW algorithm, a sliding window is applied to the backward recursion where the stakes are initialized from the previous Turbo iteration. If the window length is W, $(L/W)\times 2$ stakes needed to be saved (note that MAP 1 can only be initialized with stakes from MAP 1, not from MAP 2, resulting in twice the amount of stake memory). In the NSW algorithm, no sliding window is applied to the backward recursions. So only the stakes in the end of the recursion needed to be saved. It should be noted that the

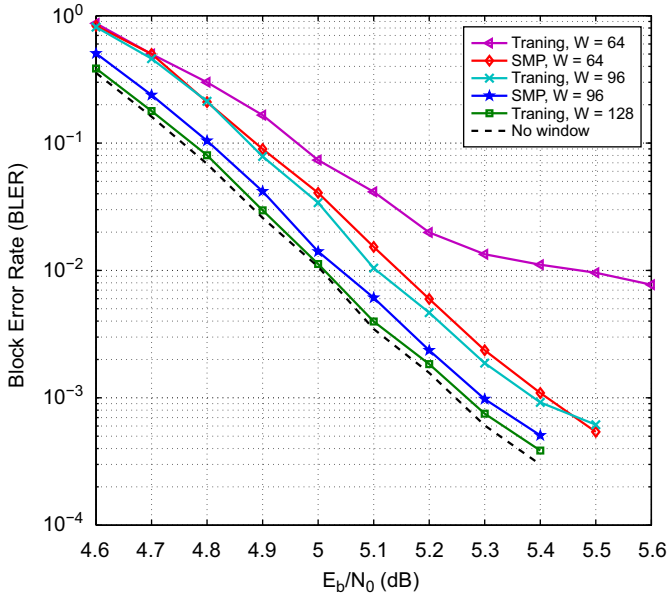

Fig. 6. Simulation result for a rate of 0.95 LTE Turbo code using two different sliding window algorithms.

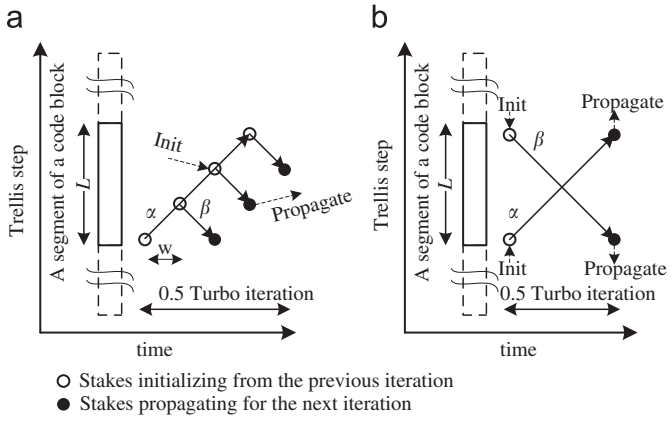

Fig. 7. Two recommended MAP decoding algorithms for LTE Turbo codes. (a) SW-MAP decoding algorithm. (b) NSW-MAP decoding algorithm.

memory bandwidth of the NSW-MAP algorithm is higher than the SW-MAP algorithm since two LLRs are read and two LLRs are written in one cycle. When the decoder parallelism is high, i.e. P is large, the NSW-MAP algorithm has throughput advantage over the SW-MAP algorithm. There are many other varieties of the MAP algorithms. See [38] for a thorough analysis of the MAP decoder architectures. In this paper, we primarily focus on these two simple but effective MAP algorithms, and we will present QPP interleaving address generator architectures for these two MAP algorithms.

4.1. QPP interleaving address generator for SW-MAP decoder

Fig. 8 shows the recommend SW-MAP decoder architecture. The SW-MAP decoder requires one set of α unit, β unit, branch unit, and LLRC unit because of the single flow structure. It employs fully parallel add-compare-select-add (ACSA) [39] units to calculate the state metrics in the α and β recursion processes. A SMP buffer was used to save the stakes for use in the next Turbo iteration. In the SW algorithm, the channel LLRs (systematic L_s and parity L_p) are loaded from the symbol memory in the sequential

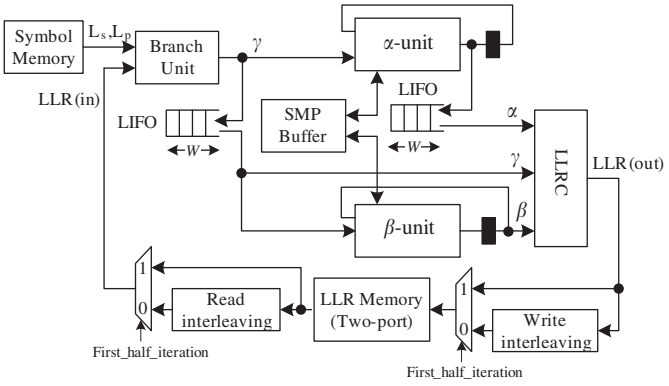

Fig. 8. SW-MAP decoder architecture.

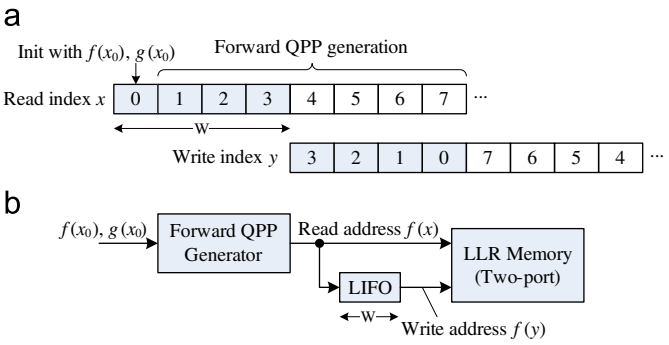

Fig. 9. (a) An example of the interleaver addressing scheme for the SW-MAP decoder, where $W=4$, $x_0=0$. (b) Architecture for generating QPP interleaving read/write addresses.

order. *A priori* information *LLR*(*in*) are loaded from the LLR memory in the sequential order for the first half iteration, and in the interleaving order for the second half iteration. The soft information *LLR*(*out*) are written to the LLR memory in the backward sequential order during the first half iteration, and in the backward interleaving order for the second half iteration. To avoid loading interleaving systematic LLR from the symbol memory during the second half iteration, we have modified the MAP algorithm to combine the systematic LLR with the extrinsic LLR in the first half iteration.

In this algorithm, the interleaving addresses must be generated during the second half iteration to provide read and write addresses to the LLR memory. In the SW algorithm, the read operation is in the forward direction, whereas the write operation is in the backward direction and is always behind of the read operation. Fig. 9(a) shows an example of the addressing scheme for $W=4$ and $x_0=0$. Fig. 9(b) shows the a hardware architecture for generating interleaving read/write addresses by using one forward QPP generator (cf. Fig. 4) and one last-in first-out (LIFO) buffer.

When the sliding window length is large, using a LIFO can be costly. We will now propose another method to generate the interleaving write addresses. As depicted in Fig. 10(b), a forward QPP address generator and a backward QPP address generator are used to recursively generate the read addresses $f(x)$ and write address $f(y)$, respectively. The initial values $f(x_0)$ and $g(x_0)$ for the forward QPP generator need to be pre-computed, whereas the initial values for the backward QPP address generator are obtained from (synchronized with) the forward QPP address generator every W cycles and then a backward recursion is performed on the next $W-1$ cycles to generate the next $W-1$

write address. Fig. 10(a) gives an example of this algorithm for $W=4$ and $x_0=0$.

4.2. QPP address generator for Radix-4 SW-MAP decoder

Radix-4 MAP decoding [13,34] is a commonly used technique to achieve a higher trellis processing speed. For binary Turbo codes, eg. LTE Turbo codes, the trellis cycles can be reduced 50% by doing Radix-4 processing. In the Radix-4 processing, during the second half iteration two LLRs for information bit vector $\{u_x, u_{x+1}\}$ are needed to be fetched/written from/to the LLR memory at addresses $f(x)$ and $f(x+1)$. Thus, two read and two write interleaving addresses need to be generated in each clock cycle. Fig. 11(a) shows an example of the read/write addressing scheme where a sequence is partitioned into even and odd sub-sequences. Fig. 11(b) shows a hardware architecture to generate the interleaving read and write addresses for the Radix-4 SW-MAP decoder. Two forward QPP address generators (with step $d=2$) are used to generate the interleaving read addresses, and two backward QPP address generators (with step $d=2$) are used to generate the interleaving write addresses. Based on the QPP algebraic property 1, the LLR memory can be partitioned into even and odd indexed banks to avoid collisions.

4.3. QPP address generator for NSW-MAP decoder

In the NSW algorithm, forward and backward recursions are performed simultaneously by processing data from both ends of the sub-trellis. After the middle point, soft LLRs are calculated in both forward and backward directions. Fig. 12 shows the NSW-MAP decoder architecture. Note that the NSW-MAP decoder requires two branch metric calculation units and two LLR calculation (LLRC) units because of the double-direction data processing. Fig. 13(a) shows the forward/backward data flow in the NSW-MAP decoding process. Because both the forward and the backward processes need to access memory, we propose to use a two phase memory accessing scheme to support double-direction data processing. As shown in Fig. 13(b), in phase 0, the

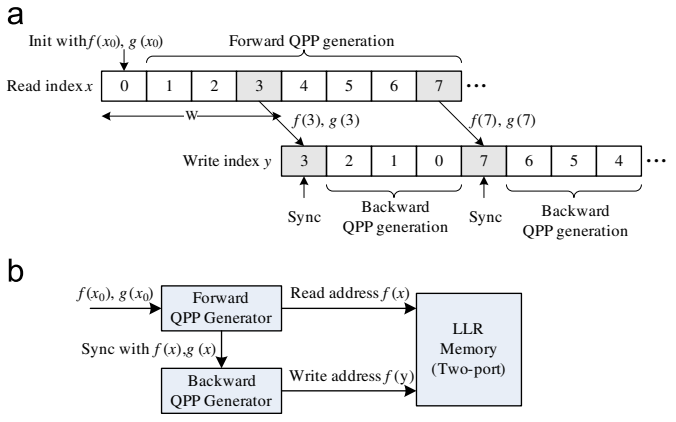

Fig. 10. (a) An example of the forward/backwoard data flow in SW-MAP algorithm, where $W=4$. (b) A hardware architecture to generate interleaving read and write addresses for SW-MAP decoder.

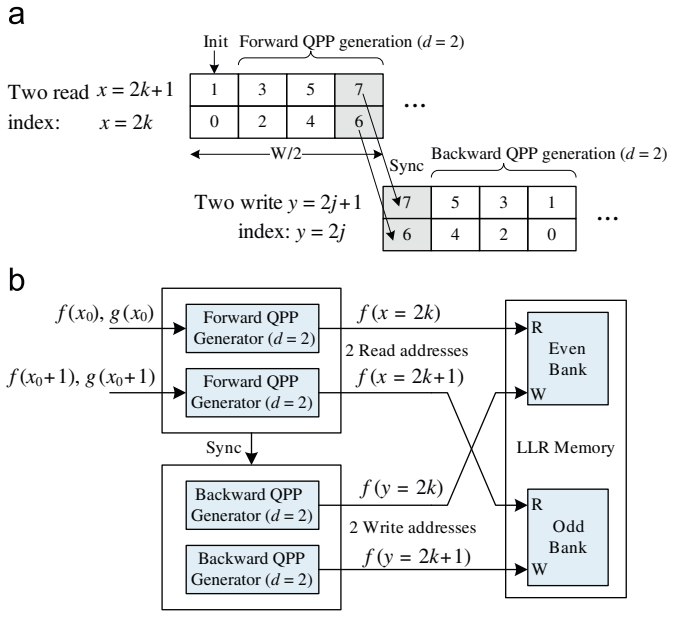

Fig. 11. (a) An example of the forward/backward data flow in Radix-4 SW-MAP algorithm, where $W=4$. (b) A hardware architecture to generate read/write interleaving address for Radix-4 SW-MAP decoder.

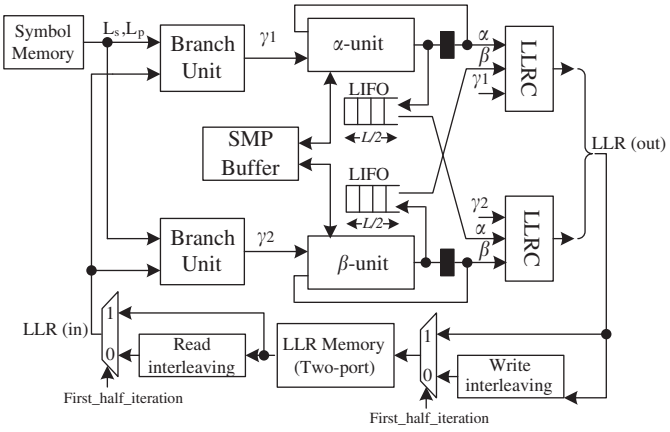

Fig. 12. NSW-MAP decoder architecture.

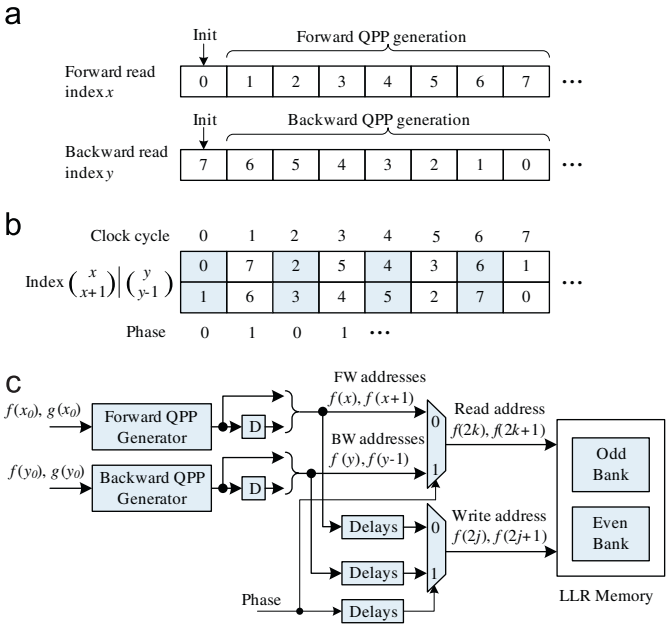

Fig. 13. (a) Forward/backward data flow in the NSW-MAP decoding process. (b) Two-phase memory accessing scheme. (c) A hardware architecture for generating interleaving addresses for the NSW-MAP decoder.

forward MAP process is allowed to read two data at addresses $f(x)$ and $f(x+1)$ from the LLR memory. In the next clock cycle (phase 1), the backward MAP process is allowed to read two data at addresses $f(y)$ and $f(y-1)$ from the LLR memory. And then this process repeats. For the write operation, it is the same as the read operation. And write address is just a delayed version of the read address. The number of the delay cycles depends on the pipeline delays in the LLRC unit in the MAP decoder which is typically several clock cycles. Fig. 13(c) shows a hardware architecture to implement this two-phase memory accessing algorithm, where the LLR memory is partitioned into even and odd indexed banks to avoid collisions.

4.4. QPP address generator for Radix-4 NSW-MAP decoder

The two-phase memory accessing scheme shown in Fig. 13(b) can be extended to support Radix-4 NSW-MAP decoding as well, where four data at addresses $f(x)$, $f(x+1)$, $f(x+2)$, and $f(x+3)$ are needed to be generated in each clock cycle. Based on the QPP algebraic property 2, the memory can be partitioned into four banks to allow concurrently memory accesses in each clock cycle without any collisions. Fig. 14 shows a hardware architecture for generating interleaving addresses for Radix-4 NSW-MAP decoder.

4.5. MAP decoder comparison

Table 2 compares the resource usage and decoding latency for a SW-MAP decoder and a NSW-MAP decoder, in which W is the sliding window length in the SW algorithm, L is the segment length $L=N/P$, B_α and B_γ are the total bit widths for the α state metrics (8 states in total) and the γ branch metrics, respectively.

To compare the area for these two types of MAP decoder architectures, we have synthesized them in a TSMC 65-nm CMOS technology for a 400 MHz clock frequency. The fixed point word lengths for the channel LLRs, extrinsic LLRs, and state metrics are 6, 7, and 10, respectively [40]. For the SW-MAP architecture, the sliding window length W is assumed to be 64. Consider decoding of a segment of a code block where the code length is $N=6144$ and the segment length is $L=N/P$, Fig. 15 shows the area cost for

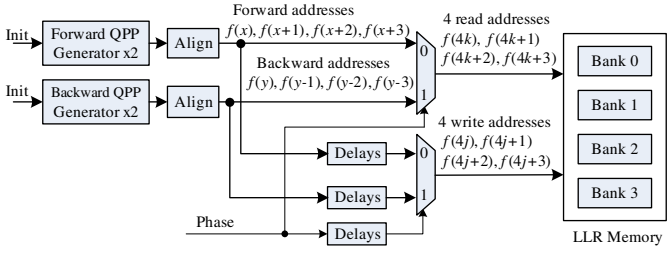

Fig. 14. A hardware architecture for generating interleaving addresses for the Radix-4 NSW-MAP decoder.

Table 2
MAP decoder architecture comparison.

	SW-MAP	NSW-MAP
α unit	1	1
β unit	1	1
Branch unit	1	2
LLRC	1	2
QPP address generator	2	2
State-buffer (bit)	$B_\alpha \times W$	$B_\alpha \times L$
γ−buffer (bit)	$B_\gamma \times W$	0
SMP-buffer (bit)	$B_\alpha \times 2L/W$	$B_\alpha \times 4$
Processing time (cycles)	$W+L$	L

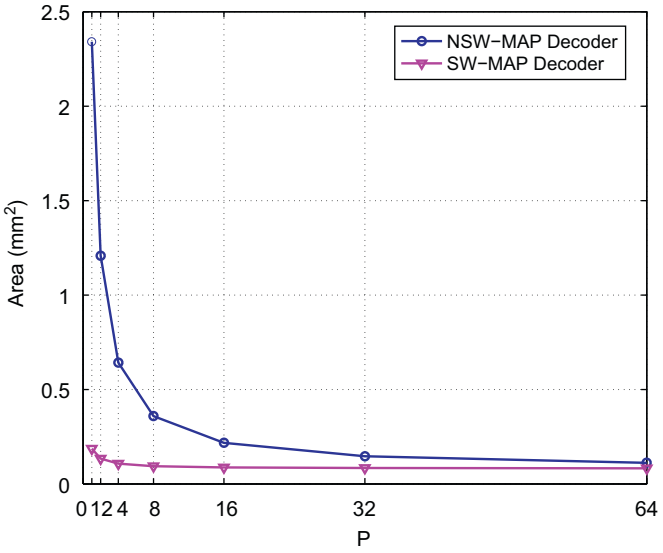

Fig. 15. Area of a NSW-MAP decoder and a SW-MAP decoder.

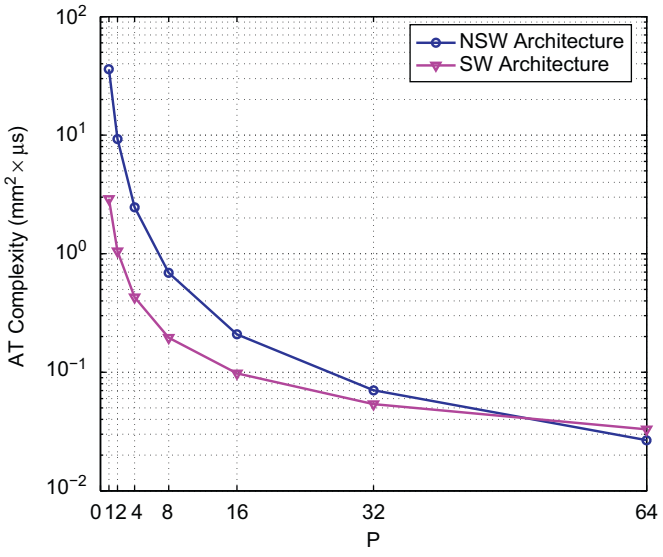

Fig. 16. AT complexity of a SW-MAP decoder and a NSW-MAP decoder.

these two types of MAP decoders. As can be seen, as the decoder parallelism P increases, the area cost of the NSW-MAP decoder reduces quickly and comes closer to the area cost of the SW-MAP decoder.

To compare the efficiency of these two architectures, we define an efficiency metric as ***area*** × ***time***, or AT, where area is one MAP decoder area and time is the processing time for a sub-trellis for half Turbo iteration. Fig. 16 plots the AT complexities for different P, where the AT value is displayed on a logarithmic scale. Clearly, when the parallelism degree P is small, the NSW-MAP architecture has a higher AT complexity than the SW-MAP architecture because a large number of state metrics have to be buffered. On the other hand, as P increases, the NSW-MAP architecture will become more efficient due to the fact that the double-flow NSW-MAP decoding has no sliding window overhead, whereas the single-flow SW-MAP decoding has a sliding window overhead of $W/(N/P+W)$. As a design tradeoff, we adopted the SW-MAP architecture in our final hardware implementation to save area while still achieving 1 Gbps throughput.

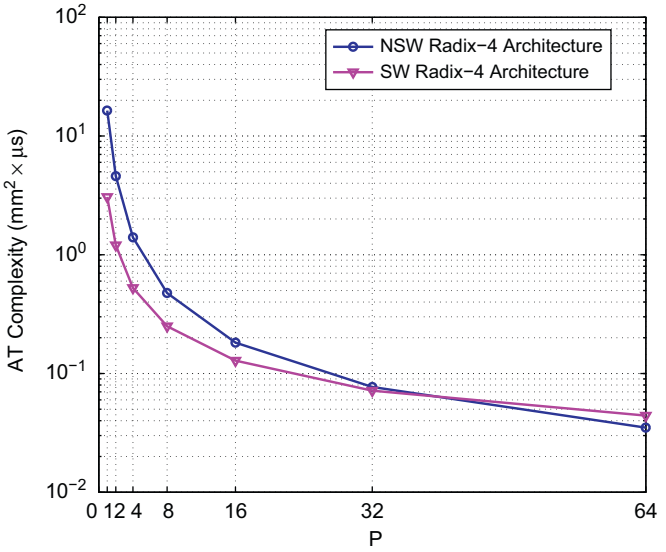

Fig. 17. AT complexity of a Radix-4 SW-MAP decoder and a Radix-4 NSW-MAP decoder.

Fig. 17 compares the AT complexities of a Radix-4 SW-MAP decoder and a Radix-4 NSW-MAP decoder for a 250 MHz clock frequency. One observation is that the Radix-4 transform can effectively reduce the AT complexity of the NSW-MAP decoder when P is small. However, Radix-4 transform will not necessarily reduce the AT complexity of the SW-MAP decoder. This is due to the fact that the Radix-2 decoder can run at a faster clock frequency, and has a lower complexity than the Radix-4 decoder (assuming full LogMAP implementation). We will compare the Radix-2 and the Radix-4 architectures in more detail in the next section.

5. Parallel turbo decoder architecture

Decoder parallelism is necessary to achieve the LTE/LTE-Advance high throughput requirement which is up to 1 Gbps. In order to increase the throughput by a factor of P, an information block can be divided into P segments with equal length L and then each segment is processed independently by a dedicated MAP decoder [14,19,33,40–47]. In this scheme, each of the P MAP cores processes the data sequentially and fetches/writes the data simultaneously always at the same offset x to each segment. The interleaver structure in the current and previous 3G standards do not have a parallel structure which makes it difficult to realize the parallelization of the MAP decoders. Expensive write buffers have to be used to reduce the memory collision caused by the interleaver [9,48]. However, when the parallelism degree increases, the collisions cannot be effectively resolved by using write buffers. The LTE QPP interleaver, however, has an inherent parallel structure that supports contention-free memory accesses which result in a large design space for the selection of appropriate levels of decoder parallelism.

In this section, we will present a scalable parallel Turbo decoder architecture and give an analysis of the complexity and the throughput. Fig. 18 illustrates the proposed parallel decoding algorithm where multiple MAP decoders are used to improve the throughput. Fig. 19 shows a hardware architecture for implementing the proposed parallel SW-MAP algorithm. In this architecture, P sets of QPP interleavers are used to generate the interleaving addresses $f(x)$, $f(x+L)$,…, and $f(x+(P-1)L)$ concurrently, where L is the segment length $L=N/P$. Based on

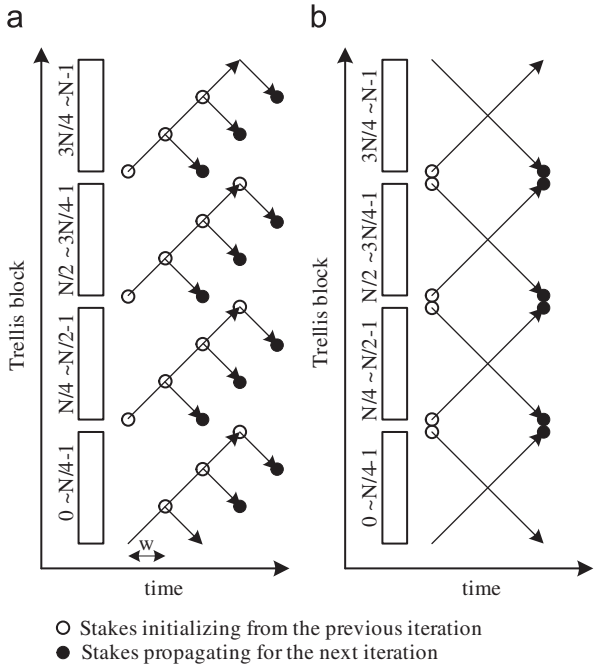

Fig. 18. An example of a multi-MAP parallel decoding approach with $P=4$. (a) Parallel SW-MAP algorithm with state metric propagation. (b) Parallel NSW-MAP algorithm with state metric propagation.

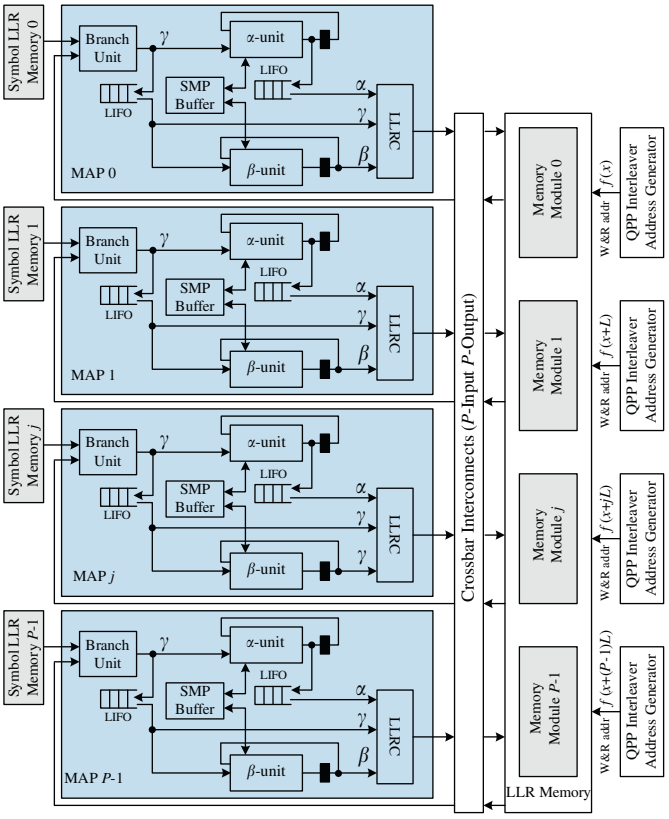

Fig. 19. The proposed parallel decoder architecture with P SW-MAP decoders. P memories are used to support contention-free memory accessing. Crossbar interconnects are used to permute the memory read/write data.

the QPP contention-free property, these P addresses will be mapped to different memory modules 0 to $P-1$ without any collisions. Thus, no write buffers are required. A crossbar network

is used to permute the data between the MAP decoders and the memory modules.

Furthermore, based on the QPP interleaver algebraic property 3, this architecture can be modified to support the Radix-4 SW and NSW MAP decoding algorithms by setting the following constraints. To support the Radix-4 SW-MAP decoding, L needs to be divisible by 2, and each memory module needs to be partitioned into even and odd indexed banks. To support the Radix-4 NSW-MAP decoding, L needs to be divisible by 4, and each memory module needs to be partitioned into four banks.

5.1. Throughput-area tradeoff analysis

High throughput is achieved by using multiple MAP decoders and multiple memory modules/banks. In this section, we will analyze the impact of parallelism on throughput and area. The maximum throughput is measured as

$$\text{SW Throughput} = \frac{N}{\text{Decoding time}} \approx \frac{N \cdot f}{I \cdot (\tilde{N}/P + \tilde{W})}$$

$$\text{NSW Throughput} = \frac{N}{\text{Decoding time}} \approx \frac{N \cdot f}{I \cdot (\tilde{N}/P)},$$

where $\tilde{N} = N$, $\tilde{W} = W$ in the case of Radix-2 decoding, and $\tilde{N} = N/2$, $\tilde{W} = W/2$ in the case of Radix-4 decoding. I is the total number of half iterations performed by the Turbo decoder. f is the operating clock frequency.

To analyze the area and throughput performance for different QPP parallelism degrees, we describe a Radix-2 and a Radix-4 SW parallel Turbo decoder in Verilog HDL and synthesize them for a 65 nm CMOS technology using Synopsys Design Compiler. The tradeoff analysis result is given in Fig. 20 which plots the area and the throughput for different parallelism degrees and clock rates. As can be seen, a 1 Gbps throughput is achievable with 64 Radix-2 MAP decoder cores running at 310 MHz clock frequency or 32 Radix-4 MAP decoder cores running at 250 MHz clock frequency.

For parallel Turbo decoder which consists of multiple MAP units, the MAP units tend to dominate the silicon area especially when the parallelism is high. From Fig. 20, we can see that given the same throughput target, the Radix-2 architecture provides a lower area cost than the Radix-4 architecture for most of the cases and especially when P is large. This is mainly due to the fact that the Radix-2 MAP unit can run at a faster clock frequency, and has a lower complexity than the Radix-4 MAP unit (assuming full LogMAP implementation). However, it should be noted that the Radix-2 decoder may need a higher partitioning of the code block than the Radix-4 decoder to achieve the same throughput target. As a design tradeoff, we adopted the Radix-2 architecture in our final hardware implementation to save area while still meeting the 1 Gbps throughput target.

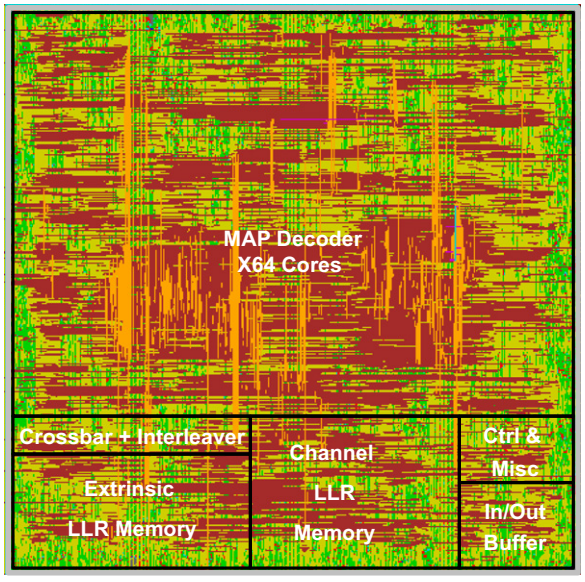

Fig. 21. VLSI layout view which shows the core area of the decoder.

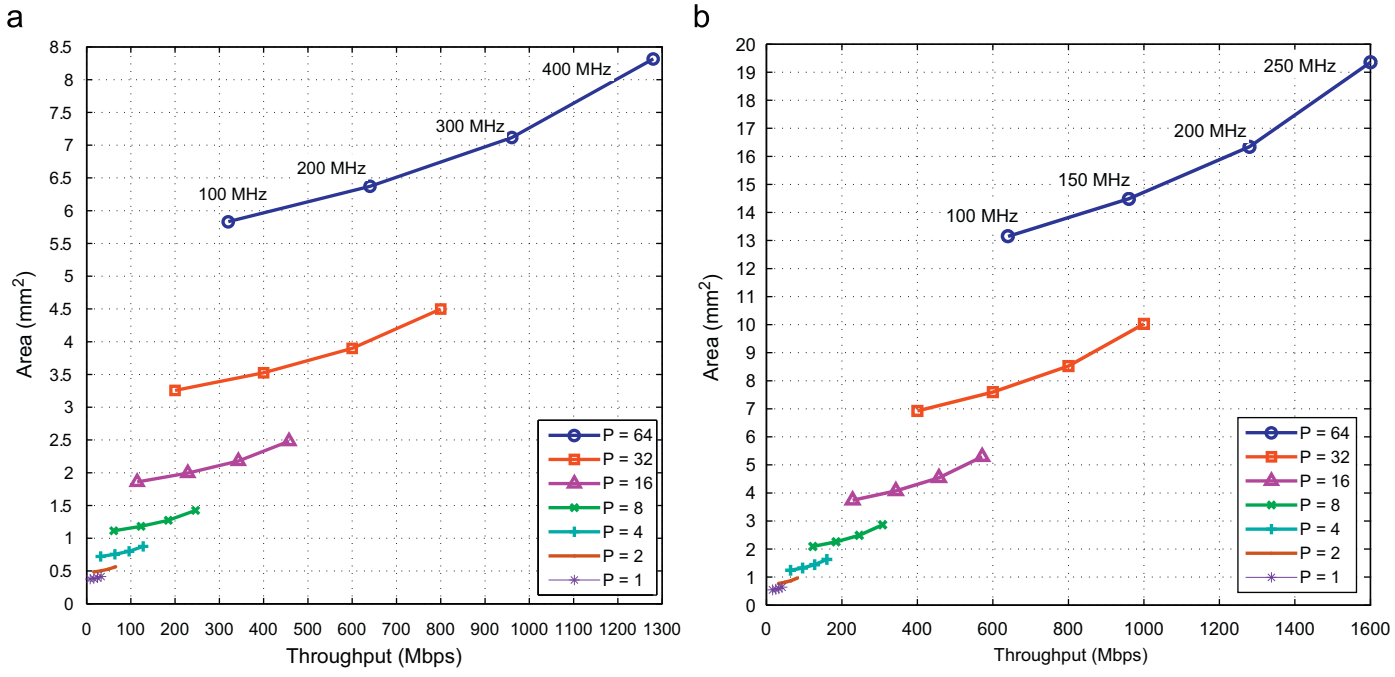

Fig. 20. Area-throughput tradeoff analysis for different parallelism and clock rates (N=6144, I=12, W=64). (a) Radix-2 SW-MAP parallel Turbo decoder. (b) Radix-4 SW-MAP parallel Turbo decoder.

Table 3
Implementation result and architecture comparison with existing Turbo decoders.

	This work	[42]	[43]	[19]	[22]
Maximum block size	6144	432	5120	6144	6144
MAP cores	64	7	6	8	8
Maximum iterations	Programmable	6	6	8	8
Technology	65 nm	180 nm	180 nm	90 nm	130 nm
Supply voltage	0.9 V	1.8 V	NA	1.0 V	1.2 V
Clock frequency	400 MHz	160 MHz	166 MHz	275 MHz	250 MHz
Core area	8.3 mm^2	7.16 mm^2	13 mm^2	2.1 mm^2	10.7 mm^2
Gate equivalent (GE)	5.8 M	587 K[a]	1.3 M[b]	740 K[c]	800 K
GE (excluding memory macros)	4.9 M	373 K	N/A	N/A	500 K
Throughput (at max. iteration)	1.28 Gbps (@6 iter.)	75.6 Mbps	60 Mbps	129 Mbps	186 Mbps
Power consumption	845 mW	N/A	N/A	219 mW	N/A
Energy efficiency (nJ/bit/iteration)	0.11	1.45	1.65	0.21	0.61

[a] The gate count is estimated based on the chip data in the paper.
[b] The unit cell area is assumed to be 10.00 μm^2 for 180 nm technology.
[c] The unit cell area is assumed to be 2.82 μm^2 for 90 nm technology.

5.2. VLSI implementation result

A highly-parallel 3GPP LTE/LTE-Advance Turbo decoder, which consists of 64 Radix-2 SW-MAP decoder cores, has been synthesized, placed and routed for a 1.0 V 8-metal layer TSMC 65 nm CMOS technology. The decoder has scalable parallelism. The decoder can employ 64, 32, and 16 MAP units when the block size $N \geq 2048$, 1024, and 512, respectively. For small block size $N < 496$, the decoder can use up to 8 MAP cores. Fig. 21 shows the top layout view of this ASIC which shows the core area of this decoder. The fixed-point bit precisions are as follows: the channel symbol LLRs for systematic and parity bits are represented with 6-bit signed numbers, the internal α and β state metrics are represented with 10-bit unsigned numbers (modulo normalization), and the extrinsic LLRs are represented with 8-bit signed numbers. Based on the fixed-point simulation result, the finite word-length implementation leads to negligible BER performance degradation from using the floating-point representation. The maximum achievable clock frequency is 400 MHz based on the post-layout simulation. The corresponding maximum throughput is 1.28 Gbps (at 6 iterations) with a core area of 8.3 mm^2.

5.3. Comparison with existing turbo decoders

In this section, we compare the proposed Turbo decoder with existing Turbo decoders from [42,43,19], and [22]. In [42], a parallel Turbo decoder based on 7 MAP decoders is presented. In order to avoid memory contention, a custom designed interleaver, which is not standard compliant, is used. In [43], a 3G-compliant parallel Turbo decoder based on the row-column permutation interleaver is introduced. In [19], a 188-mode Turbo decoder chip for 3GPP LTE standard is presented. In this decoder, 8 MAP units are used to achieve a maximum decoding throughput of 129 Mbps (at 8 iterations). In [22], a Radix-4 Turbo decoder is proposed for 3GPP LTE and WiMax standards. A maximum throughput of 186 Mbps is supported by employing 8 MAP units (at 8 iterations). Table 3 summarizes the implementation result of the proposed decoder and the hardware comparison with existing decoders. As can be seen, the proposed decoder supports 3GPP LTE-Advance throughput requirement (1 Gbps) at a small area cost, and achieves a good energy efficiency.

6. Conclusion

We have presented a highly-parallel architecture for the decoding of 3GPP LTE/LTE-Advance Turbo codes. Based on the algebraic constructions, the QPP interleaver offers contention-free memory accessing capability which enables parallel Turbo decoding by using multiple MAP decoders working concurrently. We proposed a low-complexity recursive architecture for generating the QPP interleaver addresses on the fly. The QPP interleavers are designed to operate at full speed with the MAP decoders. The proposed architecture has scalable parallelism and can be tailored for different throughput requirements. With this architecture, a throughput of 1.28 Gbps is achievable with a core area of 8.3 mm^2 in a 65-nm CMOS technology.

Acknowledgements

The authors would like to thank Nokia, Nokia Siemens Networks (NSN), Xilinx, and US National Science Foundation (under Grants CCF-0541363, CNS-0551692, CNS-0619767, EECS-0925942 and CNS-0923479) for their support of research.

References

[1] Evolved Universal Terrestrial Radio Access (EUTRA) and Evolved Universal Terrestrial Radio Access Network (EUTRAN), 3GPP TS 36.300.
[2] General UMTS Architecture, 3GPP TS 23.101 version 7.0.0, June 2007.
[3] S. Parkvall, E. Dahlman, A. Furuskar, Y. Jading, M. Olsson, S. Wanstedt, K. Zangi, LTE-advanced—evolving LTE towards IMT—advanced, in: IEEE Vehicular Technology Conference, September 2008, pp. 1–5.
[4] Multiplexing and channel coding, 3GPP TS 36.212 version 8.4.0, September 2008.
[5] C. Berrou, A. Glavieux, P. Thitimajshima, Near shannon limit error-correcting coding and decoding: turbo-codes, in: IEEE International Conference on Communication, May 1993, pp. 1064–1070.
[6] C. Berrou, A. Glavieux, Near optimum error correcting coding and decoding: turbo-codes, IEEE Transactions on Communications 44 (October) (1996) 1261–1271.
[7] L. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, IEEE Transactions on Information Theory IT-20 (March) (1974) 284–287.
[8] T.K. Blankenship, B. Classon, V. Desai, High-throughput turbo decoding techniques for 4G, in: International Conference on Third Generation Wireless and Beyond, May 2002, pp. 137–142.
[9] P. Salmela, R. Gu, S.S. Bhattacharyya, J. Takala, Efficient parallel memory organization for turbo decoders, in: Proceedings of European Signal Processing Conference, September 2007, pp. 831–835.
[10] O.Y. Takeshita, On maximum contention-free interleavers and permutation polynomials over integer rings, IEEE Trans. Inform. Theory 52 (March) (2006) 1249–1253.
[11] D. Garrett, B. Xu, C. Nicol, Energy efficient turbo decoding for 3G mobile, in: International Symposium on Low Power Electronics and Design, ACM2001, pp. 328–333.
[12] C. Chaikalis, J.M. Noras, Reconfigurable turbo decoding for 3G applications, Elsevier Signal Processing 84 (October) (2004) 1957–1972.
[13] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, C. Nicol, A 24 Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless, in: IEEE International Solid-State Circuit Conference (ISSCC), February 2003.

[14] M. Martina, M. Nicola, G. Masera, A flexible UMTS-WiMax turbo decoder architecture, IEEE Trans. Circuits and Syst. II 55 (April) (2008) 273–369.
[15] K.K. Loo, T. Alukaidey, S.A. Jimaa, High performance parallelised 3GPP turbo decoder, in: IEEE Personal Mobile Communications Conference, April 2003, pp. 337–342.
[16] M.C. Shin, I.C. Park, SIMD processor-based turbo decoder supporting multiple third-generation wireless standards, IEEE Trans. on VLSI 15 (June) (2007) 801–810.
[17] Y. Lin, S. Mahlke, T. Mudge, C. Chakrabarti, A. Reid, K. Flautner, Design and implementation of turbo decoders for software defined radio, in: IEEE Workshop on Signal Processing Design and Implementation (SIPS), October 2006, pp. 22–27.
[18] P. Salmela, H. Sorokin, J. Takala, A programmable Max-Log-MAP turbo decoder implementation, Hindawi VLSI Design 2008 (2008) 636–640.
[19] C.-C. Wong, Y.-Y. Lee, H.-C. Chang, A 188-size 2.1 mm^2 reconfigurable turbo decoder chip with parallel architecture for 3GPP LTE system, in: 2009 Symposium on VLSI Circuits, June 2009, pp. 288–289.
[20] D.-S. Cho, H.-J. Park, H.-C. Park, Implementation of an efficient UE decoder for 3G LTE system, in: International Conference on Telecommunications, June 2008, pp. 1–5.
[21] J. Berkmann, C. Carbonelli, F. Dietrich, C. Drewes, W. Xu, On 3G LTE terminal implementation—standard, algorithms, complexities and challenges, in: International Wireless Communications and Mobile Computing Conference, August 2008, pp. 970–975.
[22] J.-H. Kim, I.-C. Park, A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE, in: IEEE Custom Integrated Circuits Conference, September 2009, pp. 487–490.
[23] H.R. Sadjadpour, N.J.A. Sloane, M. Salehi, G. Nebe, Interleaver design for turbo codes, IEEE J. Sel. Areas Commun. 19 (May) (2001) 831–837.
[24] C. Schurgers, F. Catthoor, M. Engels, Optimized MAP turbo decoder, in: IEEE Signal Processing Systems (SiPS), October 2000, pp. 245–254.
[25] S.S. Pietrobon, Implementation and performance of a turbo/MAP decoder, Int. J. Satell. Commun. 16 (December) (1998) 23–46.
[26] P. Robertson, E. Villebrun, P. Hoeher, A comparison of optimal and sub-optimal MAP decoding algorithm operating in the log domain, in: IEEE International Conference on Communication, 1995, pp. 1009–1013.
[27] J. Sun, O.Y. Takeshita, Interleavers for turbo codes using permutation polynomials over integer rings, IEEE Trans. Inform. Theory 51 (January) (2005) 101–119.
[28] A. Nimbalker, K.T. Blankenship, B. Classon, T.E. Fuja, D.J. Costello, Contention-free interleavers for high-throughput turbo decoding, IEEE Trans. Commun. 56 (8) (2008) 1258–1267.
[29] A. Nimbalker, Y.W. Blankenship, B.K. Classon, K.T. Blankenship, Arp and qpp interleavers for lte turbo coding, in: IEEE Wireless Communications and Networking Conference, April 2008, pp. 1032–1037.
[30] P. Ampadu, K. Kornegay, An efficient hardware interleaver for 3G turbo decoding, in: IEEE Radio and Wireless Conference, August 2003, pp. 199–201.
[31] G. Masera, M. Mazza, G. Piccinini, F. Viglione, M. Zamboni, Low-cost IP-blocks for UMTS turbo decoders, in: 27th European Solid-State Circuits Conference, September 2001, pp. 470–473.
[32] C. Schurgers, F. Catthoor, M. Engels, Memory optimization of MAP turbo decoder algorithms, IEEE Trans. VLSI Syst. 9 (2) (2001) 305–312.
[33] S.-J. Lee, N.R. Shanbhag, A.C. Singer, Area-efficient high-throughput MAP decoder architectures, IEEE Trans. VLSI Syst. 13 (August) (2005) 921–933.
[34] Y. Zhang, K.K. Parhi, High-throughput radix-4 logMAP turbo decoder architecture, in: Asilomar Conference on Signals, Systems and Computers, October 2006, pp. 1711–1715.
[35] R. Ratnayake, A. Kavcic, G.-Y. Wei, A high-throughput maximum a posteriori probability detector, IEEE J. Solid-State Circuits 43 (2008) 1846–1858.
[36] A.J. Viterbi, An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes, IEEE J. Sel. Areas Commun. 16 (February) (1998) 260–264.
[37] J. Dielissen, J. Huisken, State vector reduction for initialization of sliding windows MAP, in: Second International Symposium on Turbo Codes and Related Topics, September 2000.
[38] M.M. Mansour, N.R. Shanbhag, VLSI architectures for SISO-APP decoders, IEEE Trans. VLSI Syst. 11 (4) (2003) 627–650.
[39] G. Masera, G. Piccinini, M. Roch, M. Zamboni, VLSI architecture for turbo codes, IEEE Trans. VLSI Syst. 7 (1999) 369–379.
[40] Y. Sun, Y. Zhu, M. Goel, J.R. Cavallaro, Configurable and scalable high throughput turbo decoder architecture for multiple 4G wireless standards, in: IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP), July 2008, pp. 209–214.
[41] Z. Wang, Z. Chi, K.K. Parhi, Area-efficient high-speed decoding schemes for turbo decoders, IEEE Trans. VLSI Syst. 10 (December) (2002) 902–912.
[42] B. Bougard, A. Giulietti, V. Derudder, J.-W. Weijers, S. Dupont, L. Hollevoet, F. Catthoor, L. Van der Perre, H. De Man, R. Lauwereins, A scalable 8.7-nJ/bit 75.6-Mb/s parallel concatenated convolutional (turbo-) codec, in: IEEE International Solid-State Circuit Conference (ISSCC), February 2003.
[43] M.J. Thul, F. Gilbert, T. Vogt, G. Kreiselmaier, N. Wehn, A scalable system architecture for high-throughput turbo-decoders, The J. VLSI Signal Process. (2005) 63–77.
[44] G. Prescher, T. Gemmeke, T.G. Noll, A parametrizable low-power high-throughput turbo-decoder, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 5, March 2005, pp. 25–28.
[45] R. Dobkin, M. Peleg, R. Ginosar, Parallel interleaver design and vlsi architecture for low-latency map turbo decoders, IEEE Trans. VLSI Syst. 13 (4) (2005) 427–438.
[46] M. May, C. Neeb, N. Wehn, Evaluation of high throughput turbo-decoder architectures, in: IEEE International Symposium on Circuits and Systems (ISCAS), May 2007, pp. 2770–2773.
[47] A. Tarable, L. Dinoi, S. Benedetto, Design of prunable interleavers for parallel turbo decoder architectures, IEEE Commun. Lett. 11 (February) (2007) 167–169.
[48] R. Asghar, D. Wu, J. Eilert, D. Liu, Memory conflict analysis and implementation of a re-configurable interleaver architecture supporting unified parallel turbo decoding, J. VLSI Signal Process. (July 2009).

Yang Sun received the B.S. degree in Testing Technology and Instrumentation in 2000, and the M.S. degree in Instrument Science and Technology in 2003, both from Zhejiang University, Hangzhou, China. From 2003 to 2004, he worked at S3 Graphics Co. Ltd. as an ASIC design engineer, developing Graphics Processing Unit (GPU) cores for graphics chipsets. From 2004 to 2005, he worked at Conexant Systems Inc. as an ASIC design engineer, developing Video decoder cores for set-top box (STB) chipsets. He is currently a Ph.D student in the Department of Electrical and Computer Engineering at Rice University, Houston, Texas. His research interests include parallel algorithms and VLSI architectures for wireless communication systems, especially forward-error correction (FEC) systems. He received the 2008 IEEE SoC Conference Best Paper Award, the 2008 IEEE Workshop on Signal Processing Systems Best Paper Award (Bob Owens Memory Paper Award), and the 2009 ACM GLSVLSI Best Student Paper Award.

Joseph R. Cavallaro, Professor
Rice University, Department of Electrical and Computer Engineering
Center for Multimedia Communication
6100 S. Main Street, MS 380, Houston, TX 77005, USA
Tel: +1 713 348 4719
Fax: +1 713 348 6196
Office: 3042 Duncan Hall
Internet: cavallar@ece.rice.edu
WWW: http://www.ece.rice.edu/~cavallar

Joseph R. Cavallaro received the B.S. degree from the University of Pennsylvania, Philadelphia, PA, in 1981, the M.S. degree from Princeton University, Princeton, NJ, in 1982, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1988, all in electrical engineering. From 1981 to 1983, he was with AT&T Bell Laboratories, Holmdel, NJ. In 1988, he joined the faculty of Rice University, Houston, TX, where he is currently a Professor of electrical and computer engineering. His research interests include computer arithmetic, VLSI design and microlithography, and DSP and VLSI architectures for applications in wireless communications. During the 1996–1997 academic year, he served at the National Science Foundation as Director of the Prototyping Tools and Methodology Program. He was a Nokia Foundation Fellow and a Visiting Professor at the University of Oulu, Finland in 2005 and continues his affiliation there as an Adjunct Professor. He is currently the Associate Director of the Center for Multimedia Communication at Rice University. He is a Senior Member of the IEEE. He was Co-chair of the 2004 Signal Processing for Communications Symposium at the IEEE Global Communications Conference and General Co-chair of the 2004 IEEE 15th International Conference on Application-Specific Systems, Architectures and Processors (ASAP).